```
~/julia$ ./julia --help
julia [switches] -- [programfile] [args...]
 -v, --version             Display version information
 -h, --help                Print this message

 -J, --sysimage <file>     Start up with the given system image file
 --precompiled={yes|no}    Use precompiled code from system image if available
 --compilecache={yes|no}   Enable/disable incremental precompilation of modules
 -H, --home <dir>          Set location of julia executable
 --startup-file={yes|no}   Load ~/.juliarc.jl
 --handle-signals={yes|no} Enable or disable Julia's default signal handlers

 -e, --eval <expr>         Evaluate <expr>
 -E, --print <expr>        Evaluate and show <expr>
 -L, --load <file>         Load <file> immediately on all processors

 --compile={yes|no|all|min}Enable or disable JIT compiler, or request exhaustive compilation
 -C, --cpu-target <target> Limit usage of cpu features up to <target>
 -O, --optimize={0,1,2,3}  Set the optimization level (default 2 if unspecified or 3 if specified as -O)
 --inline={yes|no}         Control whether inlining is permitted (overrides functions declared as @inline)
 --check-bounds={yes|no}   Emit bounds checks always or never (ignoring declarations)
 --math-mode={ieee,fast}   Disallow or enable unsafe floating point optimizations (overrides @fastmath declaration)

 --depwarn={yes|no|error}  Enable or disable syntax and method deprecation warnings ("error" turns warnings into errors)

 --output-o name           Generate an object file (including system image data)
 --output-ji name          Generate a system image data file (.ji)
 --output-bc name          Generate LLVM bitcode (.bc)
 --output-incremental=no   Generate an incremental output file (rather than complete)

 --code-coverage={none|user|all}, --code-coverage
                           Count executions of source lines (omitting setting is equivalent to "user")
 --track-allocation={none|user|all}, --track-allocation
                           Count bytes allocated by each source line
```

Both?

–> Issue #15864: separate `julia-compile` from `julia`

Julia: REPL or Compiler?

```
~/julia$ ./julia --help
julia [switches] -- [programfile] [args...]
 -v, --version             Display version information
 -h, --help                Print this message

 -J, --sysimage <file>     Start up with the given system image file
 --precompiled={yes|no}    Use precompiled code from system image if available
 --compilecache={yes|no}   Enable/disable incremental precompilation of modules
 -H, --home <dir>          Set location of julia executable
 --startup-file={yes|no}   Load ~/.juliarc.jl
 --handle-signals={yes|no} Enable or disable Julia's default signal handlers

 -e, --eval <expr>         Evaluate <expr>
 -E, --print <expr>        Evaluate and show <expr>
 -L, --load <file>         Load <file> immediately on all processors

--compile={yes|no|all|min}Enable or disable JIT compiler, or request exhaustive compilation
-C, --cpu-target <target> Limit usage of cpu features up to <target>
-O, --optimize={0,1,2,3}  Set the optimization level (default 2 if unspecified or 3 if specified as -O)
--inline={yes|no}         Control whether inlining is permitted (overrides functions declared as @inline)
--check-bounds={yes|no}   Emit bounds checks always or never (ignoring declarations)
--math-mode={ieee,fast}   Disallow or enable unsafe floating point optimizations (overrides @fastmath declaration)

--depwarn={yes|no|error}  Enable or disable syntax and method deprecation warnings ("error" turns warnings into errors)

--output-o name           Generate an object file (including system image data)
--output-ji name          Generate a system image data file (.ji)
--output-bc name          Generate LLVM bitcode (.bc)
--output-incremental=no   Generate an incremental output file (rather than complete)

--code-coverage={none|user|all}, --code-coverage
                          Count executions of source lines (omitting setting is equivalent to "user")
--track-allocation={none|user|all}, --track-allocation
                          Count bytes allocated by each source line
```

Both?

=> Issue #15864: separate `julia-compile` from `julia`

Obligatory Word-cloud

Static

Compiler

Inference

Inlining

LLVM

Parser

Source Code

Speed

Optimizations

Codegen

Efficiency

System Image

Native code

Pre-compilation

Macros

Runtime Library

Modules

Embedding

Deployment